INSTITUT DE FRANCE
Académie des sciences

Comptes Rendus
Mathématique

Pierre Auger and Olivier Pironneau

Parameter Identification by Statistical Learning of a Stochastic Dynamical System Modelling a Fishery with price variation

Volume 358, issue 3 (2020), p. 245-253

Published online: 10 July 2020

https://doi.org/10.5802/crmath.2

Les Comptes Rendus. Mathématique *sont membres du Centre Mersenne pour l'édition scientifique ouverte*
www.centre-mersenne.org
e-ISSN : 1778-3569

Comptes Rendus
Mathématique
2020, 358, nº 3, p. 245-253
https://doi.org/10.5802/crmath.2

Dynamic Programming, Statistical Learning / *Programmation dynamique, Apprentissage statistique*

Parameter Identification by Statistical Learning of a Stochastic Dynamical System Modelling a Fishery with price variation

Identification de paramètre par apprentissage statistique dans un système dynamique modélisant un site de pêche à prix variable

Pierre Auger [a] **and Olivier Pironneau** [b]

[a] IRD UMI 209, UMMISCO, Sorbonne Université, Bondy, France

[b] LJLL, Sorbonne Université, Paris 75252, cedex 5, France.

E-mails: pierre.auger@ird.fr, olivier.pironneau@sorbonne-universite.fr.

Abstract. In this short paper we report on an inverse problem for parameter setting of a model used for the modelling of fishing on the West African coast. We compare the solution of this inverse problem by a Neural Network with the more classical algorithms of optimisation and stochastic control. The Neural Network does much better.

Résumé. Dans cette courte note nous étudions les performances de l'apprentissage statistique par réseau de neurones pour l'identification des paramètres d'un modèle de pêche. L'idée est d'observer la pêche pendant quelques jours et d'en déduire les paramètres du modèle et donc la biomasse de poisson sur le long terme.

2020 Mathematics Subject Classification. 93E20.

Manuscript received 19th January 2020, accepted 20th January 2020.

1. Presentation of a fishery model with variable price

We consider a coastal area with a single fishing zone. The model introduced in [1] is summarised below.

The first equation of the model governs the fish biomass, $B(t)$, at time t, taking into account fish reproduction as well as fish mortality due to fishing. We assume that the fish population grows according to a logistic equation with a positive growth rate r and a carrying capacity K.

The mortality due to fishing is represented by a Schaefer function which is proportional to fish biomass and fishing effort $E(t)$. In case of all identical boats, the fishing effort is simply proportional to the total number of boats of the fishing fleet. The catchability q is the fishing efficiency of the fishing boats.

$$\frac{1}{B}\frac{\mathrm{d}B}{\mathrm{d}t} = r\left(1-\frac{B}{K}\right) - qE \tag{1}$$

The fishing effort is a function of the price of fish at time t, $p(t)$:

$$\frac{1}{E}\frac{\mathrm{d}E}{\mathrm{d}t} = pqB - c \tag{2}$$

This equation says that the rate of the fishing effort is proportional to the difference between the revenue and the cost of fishing per unit of fishing effort. The revenue is the catch per unit of fishing effort qB multiplied by the price p per unit of catch. The cost per unit of fishing effort c is a constant which incorporates the fuel for the boats, the taxes to pay to the government, the minimal revenue wanted by boat owners as well as the wages for the fishermen. The equation also says that if the fishery is profitable, the boat owners are going to invest in the fishery.

The price of fish on the market varies according to the difference between the demand and the supply; when the market is liquid the price adjusts quite fast (i.e. daily) to balance supply and demand. The supply is qBE while the demand is a monotone decreasing function $D(p)$ of price, for example $D(p) = \frac{A}{1+\beta p}$, where A is the carrying capacity of the market, and β a parameter. Hence

$$p = \frac{A - qBE}{\beta qBE} \tag{3}$$

For positivity of price, we assume that the catch cannot be higher than the carrying capacity of the market: $A > qBE$.

Now substituting (3) in (1) and (2) leads to:

$$\frac{\mathrm{d}B}{\mathrm{d}t} = rB\left(1-\frac{B}{K}\right) - qBE, \quad \frac{\mathrm{d}E}{\mathrm{d}t} = -cE + \frac{A}{\beta} - \frac{q}{\beta}BE \tag{4}$$

For readibility we define

$$a = \frac{r\beta}{Kq}, \quad d = \frac{Aq}{\beta}, \quad \overline{E} = qE, \quad \overline{B} = \frac{q}{\beta}B. \tag{5}$$

Accordingly,

$$\frac{\mathrm{d}\overline{B}}{\mathrm{d}t} = r\overline{B} - a\overline{B}^2 - \overline{B}\overline{E}, \quad t \in (0,T), \quad \overline{B}(0) = \overline{B}_0, \tag{6}$$

$$\frac{\mathrm{d}\overline{E}}{\mathrm{d}t} = -c\overline{E} + d - \overline{B}\overline{E}, \quad t \in (0,T), \quad \overline{E}(0) = \overline{E}_0. \tag{7}$$

The model has two variables $(\overline{B}, \overline{E})$, four positive parameters r, a, c, d and two initial conditions $\overline{B}_0$ and $\overline{E}_0$, respectively the scaled fish biomass and the scaled fishing effort at $t = 0$.

1.1. *Asymptotic limit*

The vertical axis $\overline{B} = 0$ is a nullcline and $\frac{\mathrm{d}\overline{E}}{\mathrm{d}t} = d > 0$ on the horizontal axis $\overline{E} = 0$. Therefore, the positive quadrant is positively invariant. Any trajectory starting at strictly positive initial conditions remains in the positive quadrant for ever.

We are interested by the long time limit of B: $B_\infty = \lim_{t\to\infty} B(t)$. The case $B_\infty = 0$ must be avoided because fishing will come to a dead end.

Letting $\frac{\mathrm{d}\overline{B}}{\mathrm{d}t} = \frac{\mathrm{d}\overline{E}}{\mathrm{d}t} = 0$ in (6)-(7) leads to

$$r - a\overline{B} - \overline{E} = -c\overline{E} + d - \overline{B}\overline{E} = 0 \;\Rightarrow\; a\overline{B}^2 - \overline{B}(r - ac) + d - rc = 0$$

We denote $disc = (r-ac)^2 - 4a(d-rc)$. When $disc > 0$ there are two roots

$$\bar{B}_\infty^{+,-} = \frac{1}{2a}\left(r - ac \pm \sqrt{disc}\right),\ \bar{E}_\infty^{+,-} = r - a\bar{B}_\infty^{+,-} = \frac{1}{2}\left(r + ac \mp \sqrt{disc}\right)$$

If $r > ac$ at least one of them is positive. If $rc > d$, $\bar{B}_\infty^-$ is negative.
System (6)-(7) has three equilibria $(0, d/c)$, $(\bar{B}_\infty^-, \bar{E}_\infty^-)$ and $(\bar{B}_\infty^+, \bar{E}_\infty^+)$.
The qualitative analysis shows that :

(1) If $r < d/c$ and $disc < 0$, equilibrium $(0, d/c)$ is locally asymptotically stable (l.a.s.). Any trajectory starting in the positive quadrant tends to it. In this case, the fish population goes extinct at large fishing effort.
(2) If $r < d/c$ and $disc > 0$, equilibrium $(0, d/c)$ is l.a.s., but of the two positive non trivial equilibria, $(\bar{B}_\infty^-, \bar{E}_\infty^-)$ and $(\bar{B}_\infty^+, \bar{E}_\infty^+)$, the first one with the smallest fish biomass is a saddle point and the second one is a stable equilibrium. In that case, there exists a separatrix in the phase plane. According to the initial condition, the trajectory tends either to equilibrium $(0, d/c)$, i.e. fish extinction, or to $(\bar{B}_\infty^+, \bar{E}_\infty^+)$, i.e. to a durable fishery.
(3) If $r > d/c$, equilibrium $(0, d/c)$ is unstable, there exists a single positive equilibrium $(\bar{B}_\infty^+, \bar{E}_\infty^+)$ which is l.a.s. Any trajectory starting in the positive quadrant tends to $(\bar{B}_\infty^+, \bar{E}_\infty^+)$. In this case, the fish population as well as the fishing effort tend to positive constant values. This corresponds to a durable fishery.

1.2. *Inverse problem*

To identify the parameters is difficult; but observing the system for a few days we can hope to adjust the parameters to fit the observations. The fishing effort E and the catch qBE can be observed daily easily, implying that $\bar{B}$ and $\bar{E}$ can be observed at any instant of time t_i, $i \in I$, say $\{b_i, e_i\}_{i=1,..,I}$ where I is an integer.

A brute force method is to solve the optimisation problem

$$\min_{r,a,c,d}\left\{\sum_{i\in I} |\bar{B}(t_i) - b_i|^2 + |\bar{E}(t_i) - e_i|^2 \ :\ \text{subject to (6) and (7)}\right\} \tag{8}$$

1.3. *Uncertainties*

For robustness we need to allow uncertainties on the data, and in the model, and consequently the solution of (8) is stochastic; this means that $\{\bar{B}_0, \bar{E}_0\}$ are given Gaussian random variables and to allow for uncertainty in the model we change r and d into stochastic processes $r\mathrm{d}t + \mu \mathrm{d}W_t^1$, and $d\mathrm{d}t + \mu \mathrm{d}W_t^2$. For simplicity we assume that all standard deviations are equal to μ, and all variables are uncorrelated. Let b_t, e_t be the random processes which represent $\bar{B}(t), \bar{E}(t)$. The problem is now

$$\min_{r,a,c,d} J(r,a,b,c) := \frac{1}{2}\sum_{i=1,..,I} \mathbb{E}\left[|b_{t_i} - b_i|^2 + |e_{t_i} - e_i|^2\right] \ :\ \text{subject to} \tag{9}$$

$$\mathrm{d}b_t = b_t\left((r - ab_t - e_t)\mathrm{d}t + \mu \mathrm{d}W_t^1\right), \quad b(0) = b_0 \tag{10}$$

$$\mathrm{d}e_t = (d - e_t c - b_t e_t)\mathrm{d}t + \mu \mathrm{d}W_t^2, \quad e(0) = e_0, \tag{11}$$

where W_t^1, W_t^2 are normal Gaussian processes and $\{b_0, e_0\}$ are Gaussian random variables of standard deviation μ and given means. W^1 and W^2 allow for uncertainties in the model. It is pointless to assume that b_i, e_i, $i = 1, .., I$ are random because it just shifts J as b_{t_i}, e_{t_i} are random and uncorrelated to b_i, e_i, $i = 1, .., I$.

2. Solution of the deterministic case

In absence of uncertainties, the optimisation problem (9) has 4 unkowns and easily computable gradients of the cost function with respect to these. The derivative of J with respect to one parameter, for example r, is found by letting all parameter variations to zero in the differential δJ except the variation of the one parameter set to 1, here $\delta r = 1$, $\delta a = \delta c = \delta d = 0$:

$$\delta J = \sum_{i=1,..,I} |b(t_i) - b_i|\delta b(t_i) + |e(t_i) - e_i|\delta e(t_i), \tag{12}$$

$$\frac{\mathrm{d}\delta b}{\mathrm{d}t} = b\delta r + r\delta b - b^2\delta a - 2ab\delta b - e\delta b - b\delta e, \tag{13}$$

$$\frac{\mathrm{d}\delta e}{\mathrm{d}t} = -e\delta c - c\delta e + \delta d - e\delta b - b\delta e, \tag{14}$$

2.1. *Discretization*

The time interval $[0, T]$ is split into intervals of size δt. The following semi-implicit scheme is used for $b^{m+1} \approx b((m+1)\delta t), e^{m+1} \approx e((m+1)\delta t)$:

$$e^{m+1} = (e^m + d\,\delta t)/(1 + c\,\delta t + b^m\delta t) \tag{15}$$

$$b^{m+1} = b^m(1 + r\,\delta t - e^{m+1}\delta t)/(1 + ab^m\delta t) \tag{16}$$

The scheme preserves the positivity of b, e when δt is not too large, i.e. $\delta t(r - e^{m+1}) > -1$. Hence, in case of multiple equilibrium when $t \to \infty$, it will select only the positive one.

2.2. *Numerical test*

We ran a forward problem up to $T = 10$ with $r = 2, a = 1, c = 1, d = 1.1$, with initial conditions $b_0 = 0.1, e_0 = 0.1$ and obtained $[b(t_1), e(t_1), b(t_2), e(t_2)] = [0.146, 0.307, 0.350, 0.644]$ at $t_1 = T/40, t_2 = T/10$.

Then knowing that $[b_1, e_1, b_2, e_2] = [0.146, 0.307, 0.350, 0.644]$ at $t_1 = T/40, t_2 = T/10$, we will recover the parameters $[r, a, c, d]$ up to 3 digits by using

- either the quasi-Newton method `broyden1` from the Python library `scipy`; this function requires an initial guess, here $[r, a, c, d] = [1.2, 0.5, 0.5, 0.5]$;
- or by using the least-square method `least_squares` from the same Python library. It is slightly less precise but the initial guess can be far away from the solution, here $[0.5, 0.5, 0.5, 0.5]$.

The method works equally when $[r, a, c, d] = [0.8, 1, 1, 1.1]$ which is a case $r < d/c$ and $disc < 0$, asymptotic to a biomass tending to zero.

The computing times for `broyden1` and `least_squares` are almost instantaneous on a macbook 2019, i7@2.7Ghz. However porting this strategy on a tablet or smart phone is not simple.

2.3. *Solution by statistical learning of a Neural Network*

For an introduction to Statistical Learning with Neural Networks see [3]. We have used a 4-layer Neural Network (see Figure 2). The input layer has 6 inputs; the two hidden layers have 50 neurons each and the output layer has 4 nodes. All layers use the ReLU activation. The two hidden layers are separated by a batch-normalisation. Several configurations have been tested, without much performance changes as long as the network is neither too big nor too small.

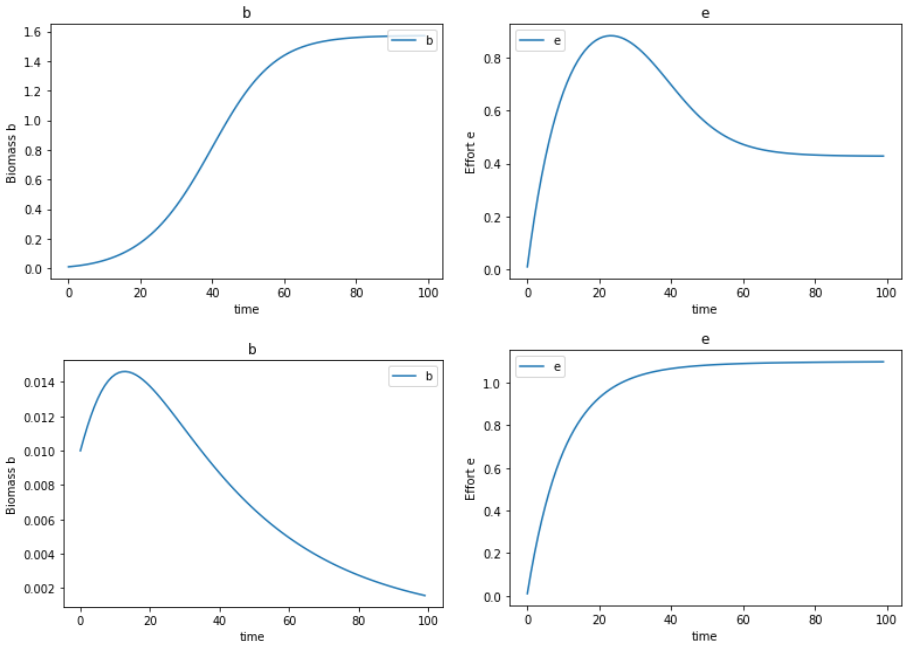

Figure 1. Evolution of the fish biomass $b(t)$ and the fishing effort $e(t)$ as a function of time, in the deterministic case when $[r,a,c,d] = [2,1,1,1.1]$. Top: when $e_0 = b_0 = 0.1$ it is a favourable case where fish biomass will not be depleted from the coast. Bottom: when $[r,a,c,d] = [0.8,1,1,1.1]$ fish biomass will disappear because $r < d/c$ and $disc < 0$.

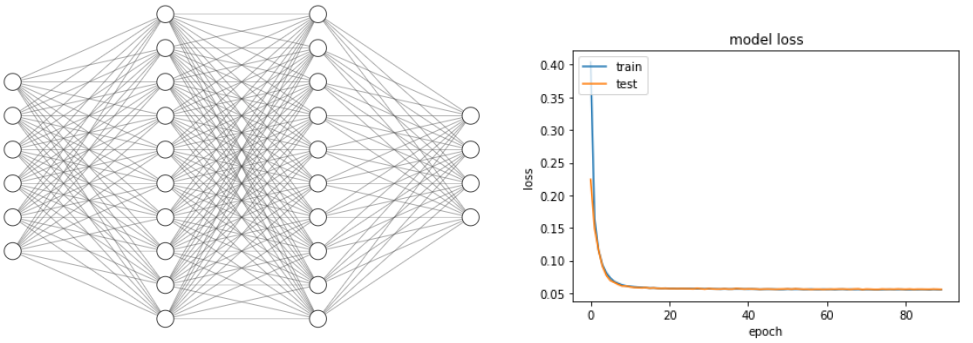

Figure 2. The Neural Network has two hidden layers each with 50 neurons and separated by batch-normalisation. Right: typical convergence curve of the loss function.

One must implement to 5 following steps.

(1) Write a Python function `fish(r,a,c,d)` which returns $[b(t_1), e(t_1), b(t_2), e(t_2)]$.
(2) Run k=1..K=5000 randomly selected cases $[r_k, a_k, c_k, d_k] \to [b_k(t_1), e_k(t_1), b_k(t_2), e_k(t_2)]$. The cases are uniformly random between 0.5 and 1.5 times the deterministic values.
(3) Set up a Neural Network which returns (output) $[r,a,c,d]$ from the knowledge (input) of $[b(t_1), e(t_1), b(t_2), e(t_2)]$. We denote by $NN([b_k(t_1), e_k(t_1), b_k(t_2), e_k(t_2)])$ the output of the Neural Network when $[b(t_1), e(t_1), b(t_2), e(t_2)]$ is the input.

(4) Train the Neural Network, i.e. adjust the Neural Network parameters by

$$\min_{param-NN} \sum_{k=1}^{K} \|NN([b_k(t_1), e_k(t_1), b_k(t_2), e_k(t_2)]) - [r_k, a_k, c_k, d_k]\|^2.$$

The parameters of the Neural Network, denoted $param - NN$ are adjusted by "`epoch<300` iterations" of the stochastic gradient method ADAM with `batch=32`.

(5) Test precision on 20 new cases $NN([b_k(t_1), e_k(t_1), b_k(t_2), e_k(t_2)])$ which should give $[r_k, a_k, c_k, d_k]$, $k > K$, for results; 9 of the 20 tests are given in Table 1.

We have implemented this algorithm using Anaconda and Keras [2].

On this test an average absolute error of $[0.0231, 0.0803, 0.0310, 0.0208]$ was obtained on $[r, a, c, d]$, which corresponds to an average relative error of $[1.2\%, 8.1\%, 3.2\%, 2.1\%]$. The loss, i.e. the value of the least square functional, at the end of the 200 iterations was 0.0976. In particular the Neural Network predicted $[2.034, 0.966, 0.995, 1.076]$ for the parameters instead of $[2, 1, 1, 1.1]$. More predicted values compared to the exact ones are shown on Table 1.

Table 1. Predicted values of the parameters by the Neural Network (left) compared with the true values (right) for the deterministic case with two time observations $t_1 = T/40, t_2 = T/10$.

r_{NN}	a_{NN}	c_{NN}	d_{NN}	r	a	c	d
1.651420	1.042426	1.342459	1.110219	1.627553	0.909005	1.359518	1.134184
2.326172	0.965113	0.543889	0.970741	2.349616	1.134547	0.575001	0.955206
2.068649	1.007492	1.280194	0.821724	2.086530	1.052534	1.284898	0.819498
1.859846	1.0944314	0.701080	0.809406	1.919166	1.343179	0.771271	0.812739
2.378332	1.152289	1.047873	1.179524	2.365285	1.285161	1.0763382	1.197219
2.230558	0.990877	0.466343	0.891779	2.182284	0.752584	0.545285	0.875104
1.788337	0.947444	0.862118	0.888756	1.800103	0.831122	0.924209	0.924773

The behaviour of the Neural Network is somewhat identical whether it is applied to the favorable case for fishing to recover $[r, a, c, d] = [2, 1, 1, 1.1]$, or the unfavourable case $[r, a, c, d] = [0.8, 1, 1, 1.1]$ (see Figure 1).

Finally we tested the same method when 3 time observations are made, at $T/40$, $T/20$ and $T/10$. The relative precision is now $[1.2\%, 7.8\%, 3.0\%, 1.5\%]$, i.e. not significantly better, and on the same test it found $[2.047, 0.963, 0.987, 1.085]$ instead of $[2, 1, 1, 1.1]$.

3. Solution of the stochastic case with a Neural Network

Assume that K cases are selected with random Gaussian values for r, c, b_0, e_0. If, for each case, we compute $b(t_i), e(t_i),\ i = 1, 2$ and then call `broyden1` to solve the inverse problem and recover the parameters and then compute their means, naturally we will find the means of r, c, b_0, e_0 which we started with.

Table 2. The random case: Relative deviation from the deterministic solution as a function of the standard deviation μ.

μ	r	a	c	d
0.125	0.046	0.20	0.09	0.03
0.25	0.07	0.19	0.19	0.08
0.5	0.09	0.19	0.17	0.08

But if we apply this strategy and solve the inverse problem with a Neural Network trained from the random parameters, i.e.: $r, c, b_0, e_0 \rightarrow b(t_i), e(t_i),\ i = 1,2$ then using the Neural Network on new cases will only produce parameters r, a, c, d, b_0, e_0 randomly near the true values. It is indeed desirable to have one trained Neural Network for all random case but then it is also difficult to evaluate its efficiency.

Otherwise there is hardly anything to change to treat the stochastic case. For every one of the K cases, Gaussian random r_t, c_t, b_0, e_0 are selected and the Neural Network training is done with these. Table 2 shows the evolution of the deviations from the deterministic answers as a function of the standard deviation μ. On Figure 3 two solutions of (9) are shown, one with $\mu = 0.125$ and the other with $\mu = 0.5$.

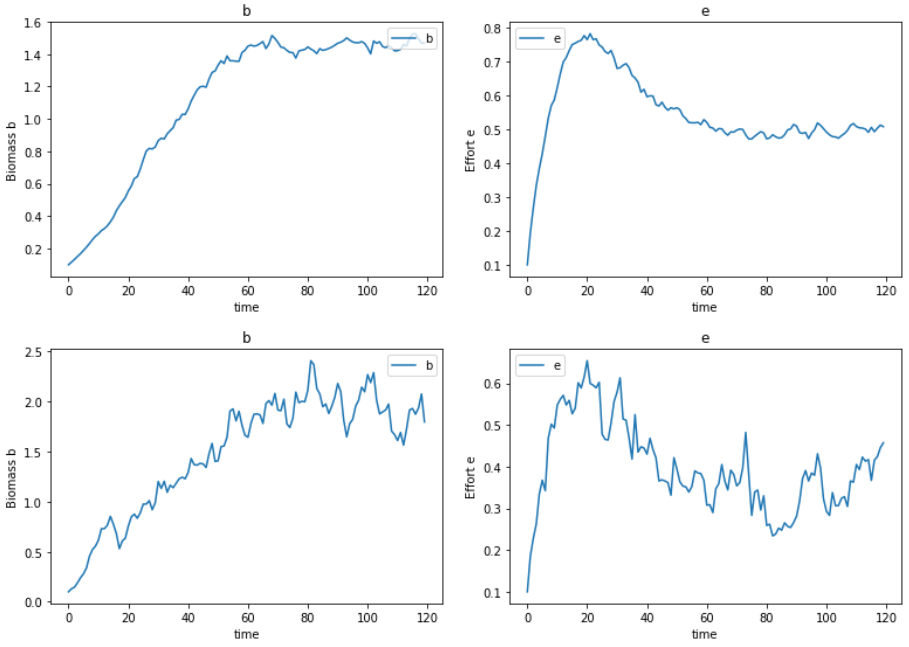

Figure 3. Response of the Neural Network to the quest for a set of parameters to fit the stochastic solution of the dynamical system with the mean of $[r, a, c, d]$ equal to $[2, 1, 1, 1.1]$ and the mean of $[b_0, e_0]$ equal to $[0.1, 0.1]$. Top: the standard deviation is $\mu = 0.125$. Bottom: $\mu = 0.5$.

4. Solution of the stochastic case by standard methods

A direct solution of the problem by a gradient method is unaffordable because it requires to solve the dynamical system a great number of time so as to compute accurately the means by a Monte-Carlo method.

Another way is to use the Kolmogorov partial differential equation for the probability density of the process $[b_t, e_t]$. However the Kolmogorov equation is very hard to solve numerically because the initial condition is a sharp Gaussian curve centred at b_0, e_0 and it is posed in an unbounded domain.

A third way is to use dynamic programming (see [5], for example), or very similarly to compute $\mathbb{E}|b_{t_i} - b_i|^2 + \mathbb{E}|e_{t_i} - e_i|^2$ by Itô calculus. Itô tells us that $\mathbb{E}|b_{t_i} - b_i|^2 + \mathbb{E}|e_{t_i} - e_i|^2$ is also $\mathbb{E}_{b_0,e_0}[u^i(b_0,e_0,0)]$ where u^i is solution of $\mathscr{P}(u^i) = 0$ where

$$\mathscr{P}(u^i) := \partial_t u^i + b(r - ab - e)\partial_b u^i + (d - ce - be)\partial_e u^i + \frac{\mu^2}{2}(b^2\partial_{bb}u^i + \partial_{ee}u^i), \tag{17}$$

$$t \in [0, t_i), \text{ with } u^i(b,e,t_i) = |b - b_i|^2 + |e - e_i|^2, \ \forall\, \{b,e\} \in \mathbb{R}^2. \tag{18}$$

A gradient based method may be used because the sensitivity of u with respect to the parameters can be computed by solving the same PDE with different right hand sides and zero conditions at t_i:

$$\mathscr{P}(u^i_{,r}) = -b\partial_b u^i,\ \mathscr{P}(u^i_{,a}) = b^2\partial_b u^i,\ \mathscr{P}(u^i_{,c}) = e\partial_e u^i,\ \mathscr{P}(u^i_{,d}) = -\partial_e u^i,$$

4.1. *Numerical implementation and Results*

There too, the fact that (17) is set in $\mathbb{R}^2$ and that $|b_{t_i} - b_i|^2 + |e_{t_i} - e_i|^2$ is not square integrable in $\mathbb{R}^2$ is a serious difficulty. One needs to localise the problem and multiply $|b - b_i|^2 + |e - e_i|^2$ by a cut-off function like $\mathbb{1}_{(0<b<1)\times(0<e<1)}$, knowing that b and e are unlikely to take greater values. A second difficulty is that it is essential to use a numerical method which provides always positive values for u. The following Eulerian–Lagrangian scheme is used to approximate (17) in time (u is generic for u^i, $i = 1,\dots,I$ and u^m approximates $u(b,e,m\delta t)$):

$$u^M(b,e) = (|b - b_i|^2 + |e - e_i|^2)\mathbb{1}_{(0<b<1)\times(0<e<1)},$$

$$\frac{1}{\delta t}\, u^m\big|_{b,e} + \frac{\mu^2}{2}(b^2\partial_{bb}u^m + \partial_{ee}u^m)\Big|_{b,e} = \frac{1}{\delta t}u^{m+1}(b - b(r - ab - e)\delta t, e - (d - ce - be)\delta t).$$

The system is localised in $(0, 2.5) \times (-3, 2)$ and discretised in space by the Finite Element Method of degree 1. Then the optimisation problem is solved using 15 iterations of the software `IPOPT`, as implemented in `FreeFem++` [4]. As for the Neural Network we impose the same constraints on the parameters:

$$1 \le r \le 3,\ 0.5 \le a \le 1.5,\ 0.5 \le c \le 1.5,\ 0.6 \le d \le 1.7.$$

Results are shown in Table 3 and on Figure 4.

Table 3. Left side: without uncertainty on b_0, e_0. Right side: with uncertainty

μ	r	a	c	d	r	a	c	d
0.01	1.953	0.743	1.474	1.464	1.806	1.5	1.5	1.455
0.125	1.760	1.027	0.648	0.854	1	1.5	1.5	1.229
0.25	1.803	1.5	1.5	1.368	1	1.5	0.5	1.524
0.5	1.702	1.5	0.5	1.558	1	1.5	0.5	1.701

For instance, for $\mu = 0.125$, IPOP found, after 15 iterations: $u = 0.0103$, $u_{,r} = 0.00985$, $u_{,a} = -0.0009738$, $u_{,c} = 0.01053$, $u_{,d} = 0.0667$ at b_0, e_0. Notice from Figure 4 that a dense grid is needed, here 100×80; the problem is numerically hard. We observe a strong dependence on the domain size, grid size and constraints on the parameters. Consequently, the results of Table 3 and not truthworthy.

We tested the method with b_0, e_0 deterministic or random. It is clear that the problem is much harder when b_0, e_0 are also random and there does not seem to be a minimum for u. Notice that a hits always its box constraint. So we tried a final test: b_0, e_0 *and* $a = 1$ fixed; when $\mu = 0.125$ the result is: $r = 1.84$, $a = 1$, $c = 1.38$, $d = 1.2$.

Acknowledgements

We would like to thank Gilles Pagès from the Sorbonne university for a valuable discussion on the Itô calculus of section 4.

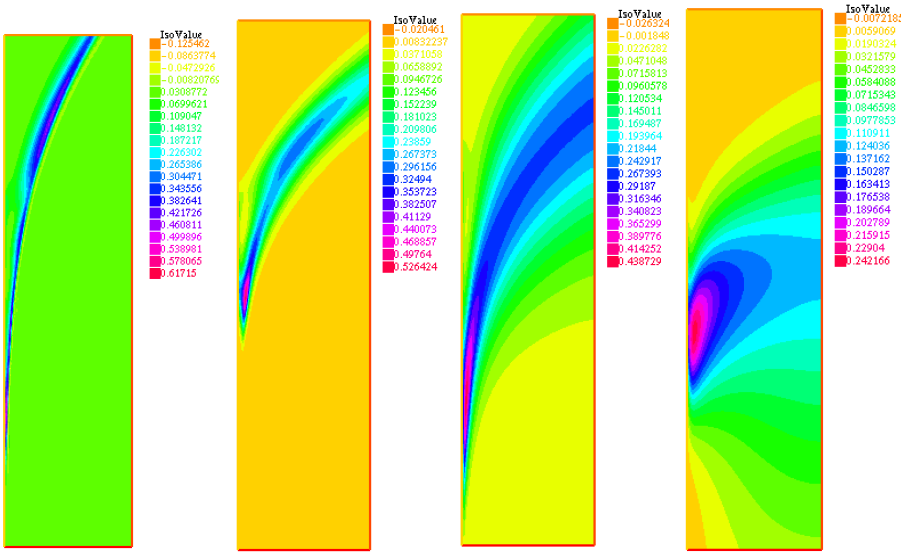

Figure 4. Color map of u in $(0,2.5)\times(-3,2)$ when the standard deviation is (from left to right) $\mu = [0.01, 0.125, 0.25, 0.5]$.

References

[1] T. Brochier, P. Auger, D. Thiao, A. Bah, S. Ly, T. N. Huu, P. Brehmer, "Can overexploited fisheries recover by self-organization? Reallocation of the fishing effort as an emergent form of governance", *Marine Policy* **95** (2018), p. 46-56.

[2] F. Chollet, *Deep learning with Python*, Manning publications, 2017.

[3] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.

[4] F. Hecht, "New development in FreeFem++", *J. Numer. Math.* **20** (2012), no. 3-4, p. 251-265.

[5] M. Laurière, O. Pironneau, "Dynamic programming for mean-field type control", *J. Optim. Theory Appl.* **169** (2016), no. 3, p. 902-924.